University of Central Florida

CAP 6412 Project Report

Bo Yang
boyang@knights.ucf.edu

March 1, 2014

This is the report of implementing Fisher vector with STIP features on UCF 101 dataset(http://crcv.ucf.edu/data/UCF101.php). In the STIP features, two low-level visual features HOG and HOF are integrated, with dimensions 72 and 90 respectively. Throughout the report, UCF 101 Split 1 is used for training and test.

1 Pipeline

The pipeline used in this project is illustrated in Figure 1. The first step is subsampling STIP features from each video clip in training list, which will be used to do PCA and train Gaussian Mixture Models(GMMs).

With the PCA coefficients and GMM parameters, treat UCF 101 video clips action by action. For each action, first load all train videos in this action(positive videos), and then randomly load the same number of video clips not in this action(negative videos). All of the loaded videos are multiplied with the saved PCA coefficients in order to reduce dimensions and rotate matrices. Fisher vectors are computed for each loaded video clip. A binary SVM model is trained with both the positive and negative Fisher vectors.

When dealing with the test videos, similar process is adopted. The only difference is that the Fisher vectors are used for SVM classification, which is based on the SVM model trained with training videos.
To well utilize the STIP features, HOG and HOF features are treated separately and they are only combined after computing Fisher vectors(concatenating HOG Fisher vectors and HOF Fisher vectors together) before SVM classification.

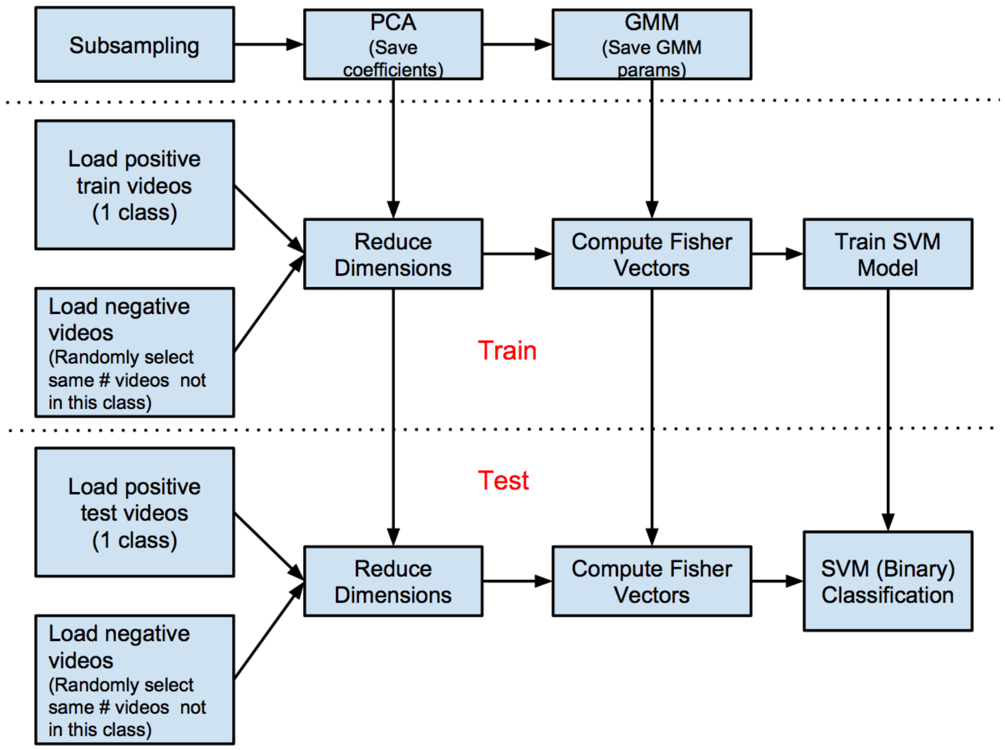
Subsampling
PCA (Save coefficients)
GMM (Save GMM params)
Load positive train videos (1 class)
Load negative videos (Randomly select same # videos not in this class)
Reduce Dimensions
Compute Fisher Vectors
Train SVM Model
Train
Test
Load positive test videos (1 class)
Load negative videos (Randomly select same # videos not in this class)
Reduce Dimensions
Compute Fisher Vectors
SVM (Binary) Classification

Figure 1: Pipeline of UCF101 action recognition using Fisher vector.

2 Pre-processing

The STIP features are stored in class, which means that all the STIP info of all video clips in each class are mixed together in a file. To extract STIP features for each video, I wrote a script(*mk_stip_data*) to separate STIP features for each video clip. And all the following operations are based on each video clip.
During the subsampling of STIP features, I randomly chose 1000 HOG or HOF features from each training video clip. For some videos, if the total number of features were less than 1000, I would use all of their features. All the subsampled features are square rooted after L1 normalization[3].

After that, the dimensions of the subsampled features were reduced to half of their original dimensions by doing PCA. At this step, the coefficients of PCA were recorded, which would be used in later. The GMMs were trained with the half-sized features, and the parameters of GMMs(i.e. weight, mean and covariance) were stored for the following process. In my program, the GMM code implemented by Oxford Visual Geometry Group(VGG,`http://www.robots.ox.ac.uk/~vgg/software/enceval_toolkit/`) is used, which eventually call VLFeat(`http://www.vlfeat.org/`). In my code, 256 Gaussians were used.

3 Fisher Vector

The implementation of Fisher vector is based on the algorithm in paper [2]. The implementation can be found in file *fisher_encode.m*6. The pre-trained GMM parameters are used as inputs, including weights, mean and covariance.

When computing the Gaussians, sometimes value *Inf* will be returned. For the *Inf* entries, a very large number(in my code, 1e30) is assigned instead to make the subsequent computation smoother. Before the L2 and power normalization, the unexpected *NaN* entries are replaced by a large number(in my implementation, 123456).

4 SVM Classification

Binary SVM classification[1] is used in my implementation. For each action, positive video clips are labeled as 1 and negative videos are as labeled -1 during training and test. In my code, SVM cost is set to 100. The option of SVM training is "*-t 0 -s 0 -q -c 100 -b 1*".

5 Results

For the first 10 actions, the mean accuracy of action recognition with STIP features is 84.32%. The confusion matrix is shown in Figure 2. Since I used binary one-vs-rest SVM classification, the incorrectly classified video clips are labeled 0, although there is no video clips are labeled 0 in UCF 101.

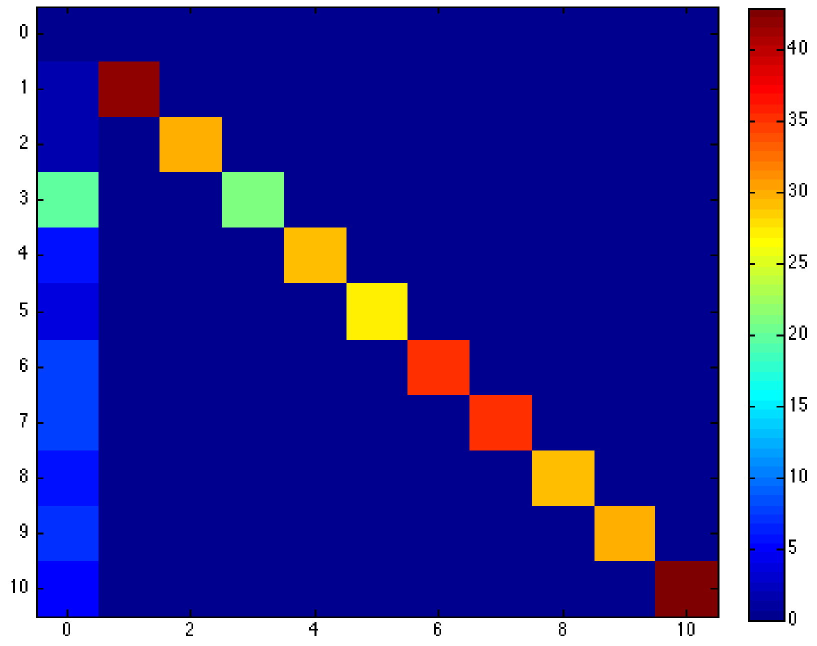

Figure 2: Confusion matrix of the first 10 actions in UCF 101.

Action 3(Archery) has the highest mis-classification rate, because the human motions in many Archery videos are very small and only a few keypoints can be detected and tracked with STIP features.

The action recognition accuracy of all the 101 actions is 77.95% when using above pipeline. And the confusion matrix is shown in Figure 3.

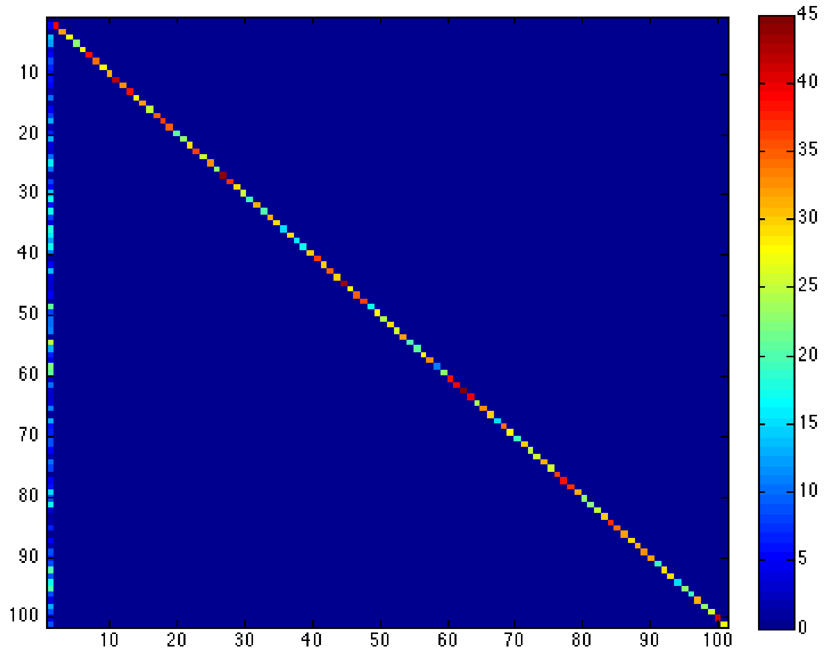

Figure 3: Confusion matrix of all the 101 actions.

6 Conclusion

In this project, the Fisher vector is implemented, and the UCF 101 dataset is tested with a one-vs-rest pipeline with the STIP features. Although my implementation got a very good result, however, due to the the limitation of STIP features, the mean accuracy is still not comparable to the state-of-the-art dense trajectory features. In general, the Fisher vector is a better encoding method than Bag-of-words for large scale data processing, and it should be used more widely in our future work.

References

[1] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[2] Jorge Sanchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image Classification with the Fisher Vector: Theory and Practice. *International Journal of Computer Vision*, 105(3):222–245, December 2013.

[3] Heng Wang and Cordelia Schmid. Action Recognition with Improved Trajectories. In *ICCV 2013 - IEEE International Conference on Computer Vision*, Sydney, Australia, December 2013. IEEE.

Appendix

Listing 1: Source code of Fisher Vector

```
function fvt=fisher_encode(feats,pca_coeff, gmm, params)
% Perform Fisher Vector encoding using PCA coefficients and pretrained GMM
% parameters.

coef=gmm.coef;
mean=gmm.mean;
var=gmm.variance;

% L1 normalization & Square root
feats=sqrt(feats/norm(feats,1));
% Apply PCA
feats=pca_coeff*feats;

D=size(feats,1);
T=size(feats,2);
K=params.K;

% Fisher vector, dimension (2D+1)*K
fvt=zeros((2*D+1)*K,1);

% Initialize accumulators
S0=zeros(K,1);
S1=zeros(K,D); % KxD matrix
S2=zeros(K,D); % KxD matrix

% Compute statistics
for t=1:T
    g=zeros(K,1); % Gaussians
    gamma=zeros(K,1); % soft assignment of x_t to Gaussian k
    for k=1:K
        % Compute the soft assignment of x_t to Gaussian k
        g(k)=Gauss(feats(:,t),mean(:,k),var(:,k)');
    end
    g(isinf(g))=1e30; % Replace inf with a very large number
    gamma=coef.*g/(coef'*g);

    S0=S0+gamma;
    S1=S1+gamma*feats(:,t)';
    S2=S2+gamma*(feats(:,t)'.^2);
end

% Compute the Fisher vector signature
Galpha=zeros(K,1);
Gmiu=zeros(K,D); % KxD matrix
Gcov=zeros(K,D); % KxD matrix

Galpha=(S0-T*coef)./sqrt(coef);
% TODO: substitute for loop by matrix multiplication
for k=1:K
    Gmiu(k,:)=(S1(k,:)-mean(:,k)'*S0(k))./(sqrt(coef(k))*var(:,k)');
    Gcov(k,:)=(S2(k,:)-2*S1(k,:)*mean(:,k)+((mean(:,k)').^2-(var(:,k)').^2 )*S0(k))./(sqrt
        (2*coef(k))*(var(:,k)').^2);
end
fvt=[Galpha;Gmiu(:);Gcov(:)];

% Replace NaN with very large number
fvt(isnan(fvt)) = 123456;

% Apply normalizations
% power normalization
fvt = sign(fvt) .* sqrt(abs(fvt));
% L2 normalization
fvt = double(fvt/sqrt(fvt'*fvt));
```

```

end

function u=Gauss(x,miu,sigma)
% Computer Gaussian distribution
%   x - 1xD input feature
%   miu - 1xD mean
%   sigma - 1xD variance

D=size(x,1);
u=1./((2*pi)^(D/2)*sqrt(abs(sigma)))*exp(-0.5*(x-miu)'*(1./sigma')*(x-miu));

end
```